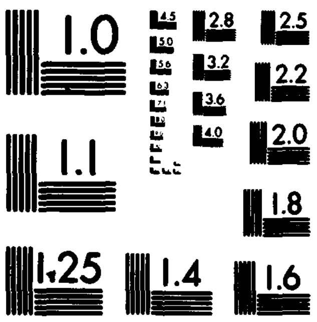


MICROCOPY RESOLUTION TEST CHART

NATIONAL BUREAU

COLORADO STATE
UNIVERSITY
FORT COLLINS, COLORADO
80523

department of electrical engineering

A Random Sampler of Dynamic Programming Applications in Signal Processing and Control

by

Louis L. Scharf

Howard Elliott

Technical Report No. DEC79-1 ✓

December 1979

14) ESL-TR-DEC79-1

11) Dec 79

9) Technical repts

6)

12)

A RANDOM SAMPLER OF DYNAMIC PROGRAMMING APPLICATIONS IN SIGNAL PROCESSING AND CONTROL

This invited paper presented at 13th Annual Asilomar Conference on Circuits, Systems, and Computers, Nov. 5-7, 1979

by

10) Louis L. Scharf Howard Elliott
~~Department~~ of Electrical Engineering
Colorado State University
Fort Collins, Colorado 80523

Abstract

I have selected three classical problems - discrete Fourier analysis, linear prediction, and nonlinear phase and frequency demodulation - to illustrate the use of dynamic programming in control and signal processing. The first two problems illustrate the use of dynamic programming in the (re-) derivation of the FFT and Levinson algorithms, two of the best known and most widely used fast algorithms. The last problem illustrates how dynamic programming can provide comfort in those applications where recursively computable, finite-dimensional sufficient statistics are nowhere to be found.

I. Introduction[2]

When I was a graduate student at the University of Washington in the late 1960s, the control students studied dynamic programming and the communication students didn't. I didn't. In the early 1970s, while covering a controls class for one of my colleagues at Colorado State University, I was forced to work through one of those classical dynamic programming problems involving *a weary salesman, a balky stationwagon,* and a sales territory as big as the whole outdoors. I remember finding the insights and techniques of dynamic programming interesting. I remember finding the problem fun to work. But I do not remember thinking, "... here is a generally applicable technique that can make my life in signal processing easier."

In the mid-1970s I read a paper by Charles Cahn [1] concerning the so-called Viterbi algorithm and its application to FM demodulation. The idea was to extend the threshold in FM by demodulating with delay, using an adaptation of dynamic programming. Slowly but surely I began to realize that traveling salesmen problems (and traveling salesmen, too) come in many guises. I now believe dynamic programming has a rôle to play in a variety of classical nonlinear filtering problems involving the demodulation of phase and frequency modulated data. This is one of the points I hope to make.

If you are a control theorist who has grown up with dynamic programming you may be thinking my experiences with dynamic programming are rather too provincial to be at all typical. Without presuming to speak for my colleagues, let me at least offer the following. In 1967 Andrew Viterbi published what was to become a prize-winning, and very influential, paper in the IT Transactions. The paper was titled "Error Bounds for Convolutional Codes and an Asymptotically Optimum Decoding Algorithm [2]". In this paper Viterbi derived an algorithm (since dubbed the Viterbi algorithm) for finding the most likely finite-state convolutional code sequence, based on noisy data. As far as I know, the Viterbi algorithm was not widely recognized as a forward dynamic programming algorithm until Forney published his account of it in 1973 [3].

[1]This work supported in part by the Army Research Office, Research Triangle Park, NC, and the Office of Naval Research, Washington, DC.

[2]In this section LLS is speaking.

II. Dynamic Programming, the DFT, and the FFT

The DFT certainly constitutes one of the cornerstones of modern Fourier analysis. Its uses range over the entire spectrum *(so to speak)* of signal processing applications. American Microsystems now markets a programmable signal processor chip with 256 16-bit words of RAM and 256 17-bit words of ROM that performs 3 million 12 bit x 12 bit multiplies each second. The chip can be programmed to perform 600 32-point complex FFTs each second. With bit-slice μcomputer architectures or special purpose random logic, one can probably realize something like 25,000 256-point complex FFTs each second. Special purpose FFT butterfly chips no doubt exist. What this means is that DFT analysis in control and signal processing problems can be carried out in real-time.

The DFT is a mapping, DFT: $\{x_n\}_0^{N-1} \to \{X_m\}_0^{N-1}$, that takes the sequence $\{x_n\}_0^{N-1}$ into the sequence $\{X_m\}_0^{N-1}$ according to the rule

$$X_m = \sum_{k=0}^{N-1} x_n W_N^{mn} \quad , \quad m=0,1,\ldots,N-1$$
$$W_N = e^{-j2\pi/N}$$

Noting that $W_N^{-mN} = 1 \ \forall m$, we may write X_m as follows:

$$X_m = \sum_{n=0}^{N-1} x_n W_N^{-m(N-n)}$$

This calculation may be viewed as the limit of the following sequence of imbedded approximations (a common dynamic programming trick):

$$X_m^{(k)} = \sum_{n=0}^{k-1} x_n W_N^{-m(k-n)} \quad , \quad k=1,2,\ldots,N$$

Note $X_m^{(k)}$ obeys the following recursion:

$$X_m^{(k+1)} = W_N^{-m} X_m^{(k)} + W_N^{-m} x_k$$
$$X_m^{(N)} = X_m \quad ; \quad X_m^{(1)} = x_0 W_N^{-m}$$

This is the so-called Goertzel algorithm for obtaining the mth DFT variable, X_m, as the output of a digital filter excited by the sequence $\{x_n\}_0^{N-1}$. The output of the filter is read at time k=N. See Figure 1.

Dynamic Programming and the Decimation-in-frequency FFT

The Goertzel algorithm is a nice dynamic programming-like solution for the DFT. However it is not efficient. Let's see if we can improve upon it. Consider $X_m^{(k)}$ for even frequency indices m=2r:

$$X_{2r}^{(k)} = \sum_{n=0}^{k-1} x_n W_N^{-2r(k-n)} \quad , \quad k=1,2,\ldots,N$$
$$= \sum_{n=0}^{k-1} x_n W_{N/2}^{-r(k-n)} \quad , \quad k=1,2,\ldots,N$$

406434

For k even (say k=2s),

$$X_{2r}^{(2s)} = \sum_{n=0}^{2s-1} x_n W_{M/2}^{-r(2s-n)}$$

$$= \sum_{n=0}^{s-1} x_n W_{N/2}^{-2(2s-n)} + \sum_{n=s}^{2s-1} x_n W_{N/2}^{-r(2s-n)}$$

$$= W_{N/2}^{-rs} \sum_{n=0}^{s-1} x_n W_{N/2}^{-r(s-n)} + \sum_{\ell=0}^{s-1} x_{\ell+s} W_{N/2}^{-r(s-p)}$$

$$= W_{N/2}^{-rs} X_{2r}^{(s)} + Y_{2r}^{(s)}$$

This shows that the 2s-point DFT approximant $X_{2r}^{(2s)}$ may be obtained from two s-point approximants. By choosing s=N/2 and continuing backwards in this way (for odd sub-indices, as well) one arrives at a backward dynamic programming derivation of the decimation-in-frequency FFT. See Figure 2 for an elementary representation of a 4-point decimation in frequency FFT.

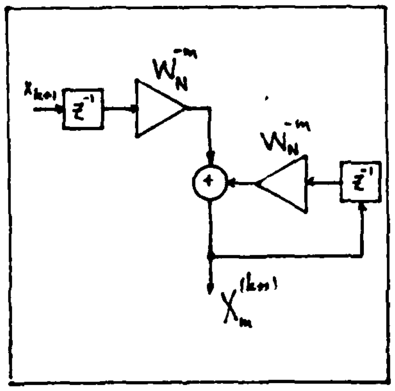


Figure 1. Goertzel Filter for DFT Component X_m

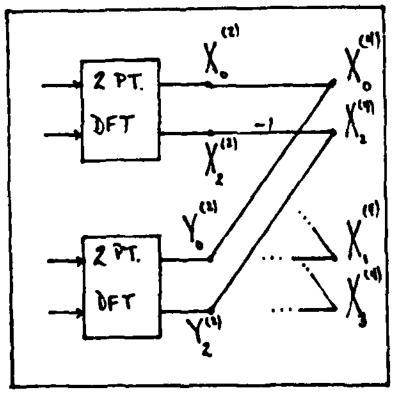
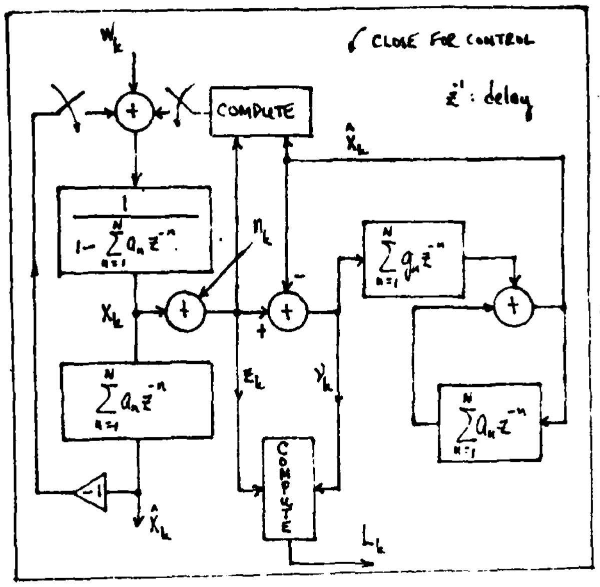


Figure 2. Four Point Decimation in Frequency FFT



CLOSE FOR CONTROL
z^{-1} : delay
W_k
COMPUTE
$\hat{X}_k$
$\frac{1}{1-\sum_{n=1}^{N} a_n z^{-n}}$
n_k
$\sum_{n=1}^{N} g_n z^{-n}$
X_k
$\sum_{n=1}^{N} a_n z^{-n}$
$\sum_{n=1}^{N} a_n z^{-n}$
z_k
y_k
-1
$\hat{X}_k$
COMPUTE
L_k

Figure 3. Prediction, Detection, and Control of a Noisy AR(N) Sequence

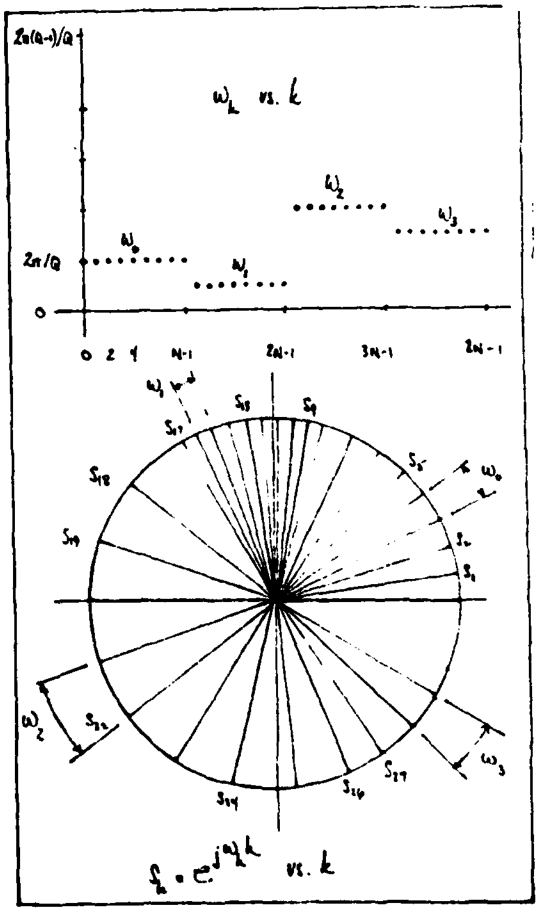


Figure 4. Visualizing the Random Walk Frequency Trajectories

III. Detection, Estimation, and Control Structures in the AR(N) Case: Kalman Filters, Levinson Recursions, and Dynamic Programming

Autoregressive (AR) models for signals, states, and data play a starring rôle in many areas of signal processing and control. By appropriately selecting model parameters (and order) one can model the covariance structure and spectral characteristics of more general models. The so-called normal equations for identifying the parameters are elegant and easily solved with recursions of the Levinson-type.

In this section we tie up control, prediction, detection, and estimation in the special case where we are dealing with a zero-mean, wide-sense stationary, scalar autoregressive time series. The usual state-variable and matrix block diagrams give way to scalar variables and digital filter blocks of moving average filters. The normal equations are high-lighted and dynamic programming is used to derive the famous Levinson recursions.

Models

Let $\{x_k\}$ denote a scalar zero-mean, wide-sense stationary autoregressive sequence that obeys the recursion

$$x_k = \sum_{n=1}^{N} a_n x_{k-n} + w_k \quad \forall\, k$$

w_n : sequence of i.i.d. $N(0,\sigma_w^2)$ r.v.s.

It is easy to see that the covariance sequence $\{r_m\}_{-\infty}^{\infty}$, $r_m = r_{-m}$, associated with the sequence $\{x_k\}$ obeys the recursion

$$r_m = \sum_{n=1}^{N} a_n r_{m-n} + \sigma_w^2 \delta_m \quad , \quad m=0,1,\ldots,$$

From here one may write out the so-called normal equations:

$$\begin{bmatrix} r_0 & r_1 & \cdots & r_{N-1} \\ r_1 & r_0 & r_1 \cdots & r_{N-2} \\ \cdot & & & \\ \cdot & & & r_1 \\ r_{N-1} & & r_1 & r_0 \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ \cdot \\ \cdot \\ a_N \end{bmatrix} = \begin{bmatrix} r_1 \\ r_2 \\ \cdot \\ \cdot \\ r_N \end{bmatrix}$$

If the $\{a_n\}_1^N$ are known these equations are used to solve for the $\{r_n\}$. Conversely, if the $\{r_n\}_0^N$ are known, these equations are used to solve for the $\{a_n\}_1^N$. In much of what follows we will assume the sequence $\{x_k\}$ is observed in zero-mean additive WGN:

$$z_k = x_k + n_k \quad \forall\, k$$

n_k : sequence of i.i.d. $N(0,\sigma_n^2)$ r.v.s.

The companion form state model for all of this follows:

$$X_{k+1} = A\, X_k + bU_k$$

$$Z_k = C'X_k$$

$$X_k = \begin{bmatrix} x_{k-N+1} \\ \cdot \\ \cdot \\ \cdot \\ x_{k-1} \\ x_k \end{bmatrix} \quad A = \left[\begin{array}{c|c} 0 & \\ & I \\ 0 & \\ \hline a_N & a_{N-1} \cdots a_0 \end{array}\right], \quad b=C=\begin{bmatrix} 0 \\ 0 \\ \vdots \\ 1 \end{bmatrix}, U_k=w_{k+1}$$

Noisy Prediction and the Kalman Predictor

The stationary Kalman one-step predictor for the noisily observed AR(N) sequence is

$$\hat{X}_{k+1} = A\, \hat{X}_k + K(z_k - \hat{x}_k)$$

$$\hat{x}_k = C'\hat{X}_k$$

where

$$K = APC(C'PC + \sigma_n^2)^{-1}$$

$$= \{k_1, k_2, \ldots, k_N\}' \qquad (1)$$

$$P = (A-KC')P(A-KC') + \sigma_n^2\, KK' + \sigma_w^2\, bb' \qquad (2)$$

The Kalman prediction sequence $\{\hat{x}_{k+1}\}$ for $\{z_{k+1}\}$ can be interpreted as the output of an ARMA(N,N-1) filter, driven by the prediction error sequence $\{\nu_k = z_k - \hat{x}_k\}$ or an ARMA(N,N-1) filter driven by the observation sequence $\{z_k\}$. The resulting filter equations are

$$\hat{x}_{k+1} = \sum_{i=1}^{N} a_i\, \hat{x}_{k+1-i} + \sum_{i=1}^{N} g_i \nu_{k+1-i}$$

or

$$\hat{x}_{k+1} = \sum_{i=1}^{N} (a_i - g_i)\hat{x}_{k+1-i} + \sum_{i=1}^{N} g_i z_{k+1-i} \qquad (3)$$

where the coefficients, g_i, can be defined by the characteristic polynomial $\Delta(\lambda)$ of $(A-KC')$:

$$\Delta(\lambda) = \lambda^N + \sum_{i=1}^{N} (g_i - a_i)\lambda^{N-i}$$

See Figure 3 for a block diagram of this predictor. Note that the noise-free MA predictor filter (next section), $\sum_{n=1}^{N} a_n z^{-n}$, is preserved in the feedback loop, but that the residual sequence $\nu_k = z_k - \hat{x}_k$ is now weighted with a feedforward MA filter, $\sum_{n=1}^{N} g_n z^{-n}$.

Why is the noisy Kalman predictor ARMA and not MA? The answer is that $\{z_k\}$, a noisy version of an AR signal process, obeys an ARMA(N,N) difference equation. As an AR(N) model has an MA(N-1) predictor, it is at least logical (if not intuitive) that an ARMA(N,N) process has an ARMA(N,N-1) predictor.

The Noise Free Predictor

The prediction vector $\hat{X}_k$ consists of the terms

$$\begin{bmatrix} E[x_{k-N+1}/z_{k-1}, \ldots] \\ \cdot \\ \cdot \\ E[x_{k-1}/z_{k-1}, z_{k-2}, \ldots] \\ E[x_k/z_{k-1}, z_{k-2}, \ldots] \end{bmatrix}$$

When $\sigma_n^2=0$, then $z_k = x_k \; \forall \; k$ and

$$E[x_{k-n}/z_{k-n}, z_{,-n-1}, \ldots]$$
$$= E[x_{k-n}/x_{k-n}, \ldots] = x_{k-n} \quad , \quad n=1,2,\ldots$$

So in this case

$$\hat{X}_{k/k-1} = \begin{bmatrix} x_{k-N+1} \\ x_{k-N+2} \\ \cdot \\ \cdot \\ \cdot \\ x_{k-1} \\ \hat{x}_{k/k-1} \end{bmatrix}$$

It follows that P, the covariance $E[\hat{X}_k - X_k][\hat{X}_k - X_k]'$ is

$$P = \begin{bmatrix} 0 & \ldots & 0 \\ \cdot & & \cdot \\ \cdot & & \cdot \\ \cdot & & 0 \\ \cdot & & \\ 0 & \ldots 0 & \sigma_u^2 \end{bmatrix} \tag{4}$$

Calculating K by substituting (4) into (1) one finds that $\Delta(\lambda) = \lambda^N$ and hence $g_i = a_i$. This implies, as one would expect, that the prediction filter (3) reduces to the purely moving average relation

$$\hat{x}_{k+1} = \sum_{i=1}^{N} a_i \, z_{k+1-i} \quad .$$

Minimum Variance Control

One of the simplest control strategies is minimum variance regulation where one desires to minimize the variance of the AR(N) output sequence $\{x_k\}$, and force $E(x_k)=0$. The well known separation principle allows one to generate a feedback control strategy assuming noisefree measurements, i.e. $n_k=0$, and then use the same strategy in the noisy case but with the Kalman filter estimates $\{\hat{x}_k\}$ replacing the actual filter outputs $\{x_k\}$.

Assume then we have the system

$$x_k = \sum_{i=1}^{N} a_i x_{k-i} + w_k + v_k$$

where $\{v_k\}$ is our feedback control sequence. We would like to minimize

$$E(x_k^2) = E\left(\sum_{i=1}^{N} a_i \, x_{k-i} + w_k + v_k\right)^2$$
$$= E\Big(w_k^2 + 2w_k v_k + 2w_k\Big(\sum_{i=1}^{N} a_i x_{k-i}\Big) +$$
$$\Big(w_k + \sum_{i=1}^{N} a_i x_{k-i}\Big)^2\Big)$$
$$= E(w_k^2) + 2E(w_k v_k | v_k)E(v_k)$$
$$+ 2E\Big(w_k\Big(\sum_{i=1}^{N} a_i x_{k-i}\Big)\Big) + E\Big(\Big(v_k + \sum_{i=1}^{N} a_i x_{k-i}\Big)^2\Big).$$

Since $\{w_k\}$ is uncorrelated with $\{x_{k-i}\}$, $i \geq 1$, and since $E(w_k v_k | v_k) = 0$, it is clear that $E(x_k^2)$ is minimized by choosing

$$v_k = -\Sigma \, a_i x_{k-i}$$

This control is illustrated in Figures 3 as a feedback loop running up the left side of the figure. The feedback loop to the top compute box shows how $\hat{x}_k$ would be used for minimum variance control in the noisy case.

Detection and the Likelihood Ratic

Consider the hypothesis test H_0 vs. H_1 with

$$H_0 : z_k = n_k \quad , \quad k=0,1,\ldots,K$$
$$H_1 : z_k = x_k + n_k \quad , \quad k=0,1,\ldots,K$$

This test is equivalent to the test $\hat{H}_0$ vs. $\hat{H}_1$ where

$$\hat{H}_0 : \nu_k^{(0)} = z_k : N(0,\sigma_n^2)$$
$$\hat{H}_1 : \nu_k^{(1)} : N(0, p_0+\sigma_n^2) \; ; \; p_0 : \text{variance of } \hat{x}_k$$

and $\nu_k^{(1)} = z_k - \hat{x}_k$ is the innovations sequence in the Kalman filter. The log-likelihood ratio for this problem is

$$LR = K - \frac{1}{p_0+\sigma_n^2} \sum_{k=0}^{K} \nu_k^2 + \frac{1}{\sigma_n^2} \sum_{k=0}^{K} z_k^2$$

Thus the statistics $\Sigma\nu_k^2$ and Σz_k^2 are sufficient and the log-likelihood ratio may be computed as in Figure 3.

The Normal Equations are Fundamental

It should be clear from Figure 3 that the AR coefficients $\{a_n\}_1^N$ characterizing the sequence $\{x_k\}$ are fundamental to the implementation of control, prediction, and detection algorithms on noisily observed AR sequences. Unfortunately, sequences rarely come tagged with their corresponding AR parameters. More typically finite records of them come to us and we estimate a covariance function (or power spectrum), often by FFT-ing, squaring and windowing, and inverse FFT-ing. These estimates may then be used to solve for the coefficients $\{a_n\}_1^N$ from the normal equations

$$\begin{bmatrix} r_0 & \cdot\;\cdot\;\cdot & r_{N-1} \\ r_1 & r_0 & \cdot \\ & & \cdot \\ & & \cdot \\ r_{N-1} & \cdot\;\cdot\;\cdot & r_0 \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ \cdot \\ \cdot \\ a_N \end{bmatrix} = \begin{bmatrix} r_1 \\ r_2 \\ \cdot \\ \cdot \\ r_N \end{bmatrix} \tag{4}$$

This makes the normal equations fundamental and arouses our interest in efficient ways of solving them.

Dynamic Programming and Levinson's Algorithm

Rewrite the normal equations as

$$\sum_{m=1}^{N} \alpha_m^N \, r_{|n-m|} = r_n \quad , \quad n=1,2,\ldots,N \tag{4a}$$

These equations characterize the $\{\alpha_m\}_1^N$ that minimize the quadratic form

$$Q_N = r_0 - 2\sum_{m=1}^{N} \alpha_m r_m + \sum_{m=1}^{N} \sum_{n=1}^{N} \alpha_m \alpha_n \, r_{|n-m|}$$

The minimum is

$$Q_N^N(r_1,\ldots,r_N) = r_0 - \sum_{m=1}^{N} \alpha_m^N \, r_m \tag{4b}$$

Assuming the normal equations to be non-singular, we may write

$$\alpha_m^N = \sum_{n=1}^{N} S_{mn}^N r_n \quad , \quad m=1,2,\ldots,N \tag{5}$$

$$Q_N^N(r_1,\ldots,r_N) = r_0 - \sum_{m=1}^{N} \sum_{n=1}^{N} r_m S_{mn}^N r_n \tag{6}$$

Write out $Q_N(\cdot)$ as follows:

$$Q_N(r_1,\ldots,r_N) = r_0 - 2\sum_{m=1}^{N-1} \alpha_m r_m - 2\alpha_N r_N + \sum_{m=1}^{N-1}\sum_{n=1}^{N-1} \alpha_m \alpha_n r_{|m-n|}$$
$$+ 2\sum_{m=1}^{N-1} \alpha_m \alpha_N r_{|N-m|} + \alpha_N^2 r_0$$
$$= \alpha_N^2 r_0 - 2\alpha_N r_N + Q_{N-1}(r_1 - \alpha_N r_{N-1},\ldots,r_{N-1}-\alpha_N r_1)$$

So the minimization of $Q_N(r_1,\ldots,r_N)$ with respect to $\{\alpha_m\}_1^N$ leads to

$$Q_N^N(r_1,\ldots,r_N) = \min_{\{\alpha_m\}_1^N} Q_N(r_1,\ldots,r_M)$$
$$= \min_{\alpha_N}[\alpha_N^2 r_0 - 2\alpha_N r_N + \min_{\{\alpha_m\}_1^{N-1}} Q_{N-1}(r_1 - \alpha_N r_{N-1},\ldots)]$$
$$= \min_{\alpha_N}[\alpha_N^2 r_0 - 2\alpha_N r_N + Q_{N-1}^{N-1}(r_1 - \alpha_N r_{N-1},\ldots)] \tag{7}$$

<u>This equation contains the essence of dynamic programming and the principle of optimality:</u> Once the $\{\alpha_1,\ldots,\alpha_{N-1}\}$ are found, Q_{N-1}^{N-1} may be constructed and α_N found as a function of r_0, r_N and $\{\alpha_1,\ldots,\alpha_{N-1}\}$. One continues in this way. At each step of the way the minimization problem on $Q_{N-1}(\cdot)$ is quadratic.

Let's use (6) in the RHS of (7):

$$Q_N^N(r_1,\ldots,r_N) = \min_{\alpha_N}[\alpha_N^2 r_0 - 2\alpha_N r_N + r_0$$
$$- \sum_{m=1}^{N-1}\sum_{n=1}^{N-1}(r_m - \alpha_N r_{N-m}) S_{mn}^{N-1}(r_n - \alpha_M r_{M-n})]$$
$$= \min_{\alpha_N}[\alpha_N^2(r_0 - \sum_{m=1}^{N-1}\sum_{n=1}^{N-1} r_{N-m} S_{mn}^{N-1} r_{N-n})$$
$$-2\alpha_N(r_N - \sum_{m=1}^{N-1} r_m \sum_{n=1}^{N-1} S_{mn}^{N-1} r_{N-n})$$
$$+ Q_{N-1}^{N-1}(r_1,\ldots,r_{N-1})]$$

It follows easily that the minimizing value of α_N is

$$\alpha_N^N = \frac{r_N - \sum_{m=1}^{N-1}\sum_{n=1}^{N-1} r_m S_{mn}^{N-1} r_{M-n}}{r_0 - \sum_{m=1}^{N-1}\sum_{n=1}^{N-1} r_{N-m} S_{mn}^{N-1} r_{N-n}}$$

Use (5) in the numerator to get

$$\alpha_N^N = \frac{r_N - \sum_{n=1}^{N-1} \alpha_n^{N-1} r_{N-n}}{r_0 - \sum_{m=1}^{N-1}\sum_{n=1}^{N-1} r_{N-m} s_{mn}^{N-1} r_{N-n}}$$

Let's call

$$C_n^{N-1} = \sum_{m=1}^{N-1} r_{N-m} s_{mn}^{N-1} \quad , \quad n=1,2,\ldots,N-1$$

and see if we can find a recursion for it. In the meantime

$$\alpha_N^N = \frac{r_N - \sum_{n=1}^{N-1} \alpha_n^{N-1} r_{N-n}}{r_0 - \sum_{n=1}^{N-1} C_n^{N-1} r_{N-n}} \tag{8}$$

Note

$$\alpha_N^{N^2}(r_0 - \sum_{n=1}^{N-1} C_n^{N-1} r_{N-n}) - 2\alpha_N^N(r_N - \sum_{n=1}^{N-1} \alpha_n^{N-1} r_{N-n})$$
$$= \alpha_N^N(r_N - \sum_{n=1}^{N-1} \alpha_n^{N-1} r_{N-n}) - 2\alpha_N^N(r_N - \sum_{n=1}^{N-1} \alpha_n^{N-1} r_{N-n})$$
$$= -\alpha_N^{N_1}(r_N - \sum_{n=1}^{N-1} \alpha_n^{N-1} r_{N-n})$$

So we have this recursion for the minimum prediction error:

$$Q_N^N(r_1,\ldots,r_N) = Q_{N-1}^{N-1}(r_1,\ldots,r_{N-1}) - \alpha_N^N(r_N - \sum_{n=1}^{N-1} \alpha_n^{N-1} r_{N-n})$$

Now use (4b) to get

$$r_0 - \sum_{m=1}^{N} \alpha_m^N r_m = r_0 - \sum_{m=1}^{N-1} \alpha_m^{N-1} r_m - \alpha_N^N r_N + \sum_{n=1}^{N-1} \alpha_N^N \alpha_N^{N-1} r_{N-n}$$

Or

$$\sum_{m=1}^{N-1} \alpha_m^N r_m = \sum_{m=1}^{N-1} (\alpha_m^{N-1} - \alpha_N^N \alpha_{N-m}) r_m$$

Thus from the recursion on Q_N^N we get this recursion for the α_m^N:

$$\alpha_m^N = \alpha_m^{N-1} - \alpha_N^N \alpha_{N-m}^{N-1} \quad , \quad m=1,2,\ldots,M-1$$

Our one remaining problem is C_n^N. Write it as

$$C_m^N = \sum_{n=1}^{N} s_{mn}^N r_{N+1-n}$$

Thus C_m^N must be the solution of

$$\sum_{m=1}^{N} C_m^N r_{|n-m|} = r_{N+1-n} \quad , \quad n=1,2,\ldots,N-1 \tag{9}$$

It therefore satisfies a recursion just like α_m^N. In fact (9) is just (4a) with the RHS vector turned upside down. By the symmetry of $R = \{r_{|m-n|}\}$, this simply turns the solution vector $\{\alpha_m^N\}$ upside down. Thus

$$C_m^N = \alpha_{N+1-m}^N \quad , \quad m=1,2,\ldots,N$$

and we are done.

<u>Summary:</u> The Levinson recursions may now be written

$$\alpha_N^N = \frac{r_N - \sum_{n=1}^{N-1} \alpha_n^{N-1} r_{N-n}}{r_0 - \sum_{n=1}^{N-1} \alpha_{N-n}^{N-1} r_{N-n}}$$

$$\alpha_m^N = \alpha_m^{N-1} - \alpha_N^N \alpha_{N-m}^{N-1} \quad , \quad m=1,2,\ldots,N-1$$

$$Q_N^N = Q_{N-1}^{N-1} - \alpha_N^N (r_N - \sum_{m=1}^{N-1} \alpha_m^{N-1} r_{N-m})$$

To get these equations into their fully modern form, one must look at a backwards prediction.

IV. Frequency Tracking and Dynamic Programming

Phase and frequency tracking problems comprise some of the most nettlesome nonlinear filtering problems in the entire realm of signal processing and communication. A typical problem is the following: observe the sequence $\{z_k\}$ with

$$z_k = s_k + n_k$$

$$s_k = e^{j\omega_k k}$$

and estimate the FM sequence $\{\omega_k\}$. Here $s_k = e^{j\omega_k k}$ is a randomly frequency modulated signal. The sequence $\{n_k\}$ is additive noise and $\{\omega_k\}$ is a sequence of angular frequencies. Assume n_k:$N(0,\sigma^2)$ (complex normal).

We wish to observe a record of data $\{z_k\}_0^{KN}$ and infer the most likely sequence of frequencies, $\{\hat{\omega}_k\}_0^{KN}$. For our model of $\{\omega_k\}$ we take each $\omega_k \epsilon \{0, 2\pi/Q, \ldots, 2\pi(Q-1)/Q\}$, evolving according to

$$\omega_{kN} = \omega_{(k-1)N} + \nu_{kN}$$

$$\omega_{kN+\ell} = \omega_{kN} \quad , \ \ell=0,1,\ldots,N-1 \quad \forall\ k$$

A typical sequence $\{\omega_k\}$ is illustrated in Figure 4. The independent increments sequence $\{\nu_{kN}\}$ is a sequence of i.i.d. r.v.s. selected in such a way that the transition probabilities

$$f(\omega_{kN}/\omega_{(k-1)N})$$

correspond to our notion of physical reality. Physically we may think of the sequence $\{\omega_{kN}\}$ as a finite-state random walk on the circle with an unusual transition probability structure. Typical trajectories for $\{\omega_k\}$ and $\{s_k = e^{j\omega_k k}\}$ are illustrated in Figure 4.

Let's organize the sequence $\{z_k\}_0^{KN}$ into contiguous blocks of length N, of the form $\{z_{kN+\ell}\}_{\ell=0}^{N-1}$. See Figure 5. Recall

$$z_{kN+\ell} = e^{j\omega_{kN+\ell}(kN+\ell)} + n_{kN+\ell}$$

$$= e^{j\omega_{kN}(kN+\ell)} + n_{kN+\ell}$$

Write $\{z_k\}_0^{KN} = \bigcup_{k=0}^{K-1} \{z_{kN+\ell}\}_{\ell=0}^{N-1}$. Consider the joint density of $\{z_k\}_0^{KN}$ and $\{\omega_{kN}\}_0^{K-1}$:

$$f(\bigcup_{k=0}^{K-1} \{z_{kN+\ell}\}_0^{N-1}, \{\omega_{kN}\}_0^{K-1})$$

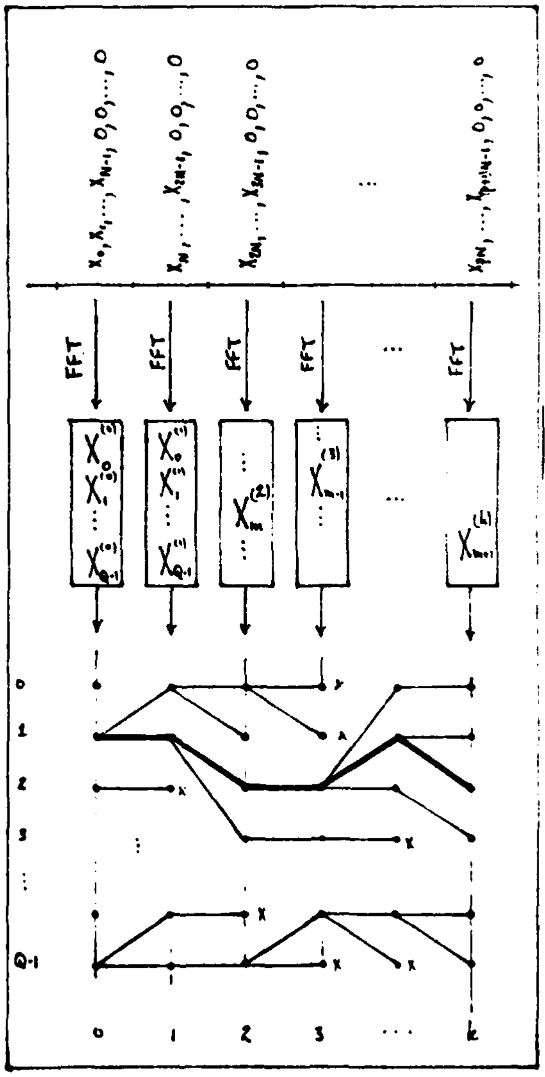
Exploit the structure of $\{z_k\}$ and $\{\omega_{kN}\}$ to write this as

$$f(\cdot,\cdot) = \prod_{k=0}^{K-1} \prod_{\ell=0}^{N-1} N_{z_{kN+\ell}}(e^{j\omega_{kN}(kN+\ell)}, \sigma_n^2) \prod_{k=0}^{K-1} f(\omega_{kN}/\omega_{(k-1)N})$$

The natural logarithm of $f(\cdot,\cdot)$ is proportional to

$$\ell n f(\cdot,\cdot) \sim 2\mathrm{Re}\ \frac{1}{2\sigma_n^2} \sum_{k=0}^{K-1} \sum_{\ell=0}^{N-1} z_{kN+\ell}\, e^{-j\omega_{kN}(kN+\ell)}$$

$$+ \sum_{k=0}^{K-1} \ell n\ f(\omega_{kN}/\omega_{(k-1)N})$$



x0, x1, ..., xN-1, 0, 0, ..., 0
xN, ..., x2N-1, 0, 0, ..., 0
x2N, ..., x3N-1, 0, 0, ..., 0
xkN, ..., x(k+1)N-1, 0, 0, ..., 0
FFT
FFT
FFT
FFT
FFT
0
1
2
3
Q-1
0
1
2
3
k

Figure 5. Data Processing and Frequency Trellis Illustrating Evolution of Surviving Frequency Tracks

$$\ell nf(\cdot,\cdot)\ 2\mathrm{Re}-\frac{1}{2\sigma_n^2}\sum_{k=0}^{K-1} e^{-j\omega_{kN}kN}\sum_{\ell=0}^{N-1} z_{kN+\ell}e^{-j\omega_{kN}\ell}+$$

$$+\sum_{k=0}^{N-1}\ell nf(\omega_{kN}/\omega_{(k-1)N})$$

Let $X_{kN}(\theta)$ denote the finite Fourier transform

$$X_{kN}(\theta)=\sum_{\ell=0}^{N-1} z_{kN+\ell}\,e^{j\theta\ell}$$

Then the log-likelihood function may be written

$$\ell nf(\cdot,\cdot)\text{-}2\mathrm{Re}-\frac{1}{2\sigma_n^2}\sum_{k=0}^{K-1} e^{-j\omega_{kN}kN}\,X_{kN}(\theta=\omega_{kN})$$

$$+\sum_{k=0}^{N-1}\ell nf(\omega_{kN}/\omega_{(k-1)N})$$

Our notion of the most likely sequence $\{\hat{\omega}_k\}_0^{KN}$ is the sequence

$$\hat{\omega}_{kN+\ell}=\hat{\omega}_{kN}\ ,\ \ell=0,1,2,\ldots,N-1\ \ ,\ k=0,1,\ldots,K-1$$

where $\{\hat{\omega}_{kN}\}_0^{K-1}$ is the sequence that maximizes $f(\cdot,\cdot)$. Thus we consider the maximization problem

$$\max_{\{\omega_{kN}\}_0^{K-1}}\ -\frac{1}{\sigma_n^2}\,\mathrm{Re}\sum_{k=0}^{K-1} e^{-j\omega_{kN}kN}\,X_{kN}(\theta=\omega_{kN})$$

$$+\sum_{k=0}^{K-1}\ell nf(\omega_{kN}/\omega_{(k-1)N})$$

Write this as

$$\max_{\{\omega_{kN}\}_0^{K-1}}\ \Gamma_{K-1}$$

with Γ satisfying the following recursion:

$$\Gamma_t=\Gamma_{t-1}+\ell nf(\omega_{tN}/\omega_{(t-1)N})+\frac{1}{\sigma_n^2}\,\mathrm{Re}\ e^{-j\omega_{tN}tN}X_{tN}(\theta=\omega_{kN})$$

So our maximization problem becomes

$$\max_{\{\omega_{kN}\}_{K-2}^{K-1}}\ \Big[\max_{\{\omega_{kN}\}_0^{K-3}}\ \Gamma_{K-2}\ \ell nf(\omega_{(K-1)N}/\omega_{(K-2)N})+$$

$$+\frac{1}{\sigma_n^2}\,\mathrm{Re}\ e^{-j\omega_{(K-1)N}(k-1)N}\,X_{(K-1)N}(\theta=\omega_{(K-1)N})$$

This form leads to the following observation: the maximizing frequency trajectory, call it $\{\hat{\omega}_{kN}\}$, passing through $\hat{\omega}_{(K-2)N}$ on its way to $\omega_{(K-1)N}$, must arrive at $\hat{\omega}_{(K-2)N}$ along a route $\{\hat{\omega}_{kN}\}_0^{K-3}$ that maximizes Γ_{K-2}. If it did not we could retain $\hat{\omega}_{(K-2)N}$ and $\hat{\omega}_{(K-1)N}$ and with a different sequence to get a larger Γ_{K-1}. It is this observation that forms the basis of forward dynamic programming.

Recall the $\omega_{kN}\ \epsilon\{2\pi r/Q\}_{r=0}^{Q-1}$. This means the finite Fourier transform $X_{kN}(\theta)$ must only be evaluated on $\theta=2\pi r/Q$, $r=0,1,\ldots,Q-1$. The best way to do this is to zero-pad $\{x_{kN+\ell}\}_{\ell=0}^{N-1}$ to obtain a Q-point sequence that can be FFT'd to get $X_{kN}(\theta=2\pi r/Q)$. See Figure 5. Then for each node on Figure 5 we evaluate $X_{kN}(\theta=2\pi r/Q)$, and find the best route through the trellis with a dynamic programming algorithm. This completes our algorithm for moderating the usual peak-picking rule on the FFT with prior information $f(\omega_{kN}/\omega_{(k-1)N})$.

References

[1] C. R. Cahn, "Phase Tracking and Demodulation with Delay," IEEE Trans. Inform. Theory, IT-20, pp. 50-58 (January 1974).

[2] A. J. Viterbi, "Error Bounds for Convolutional Codes and an Asymptotically Optimum Decoding Algorithm," IEEE Trans. Inform. Theory, IT-13, pp. 260-269 (April 1967).

[3] G. D. Forney, Jr., "The Viterbi Algorithm", Proc. IEEE, 61, pp. 268-278 (March 1973).